Microcylinder Appearance Model for Cloth Rendering Implementation

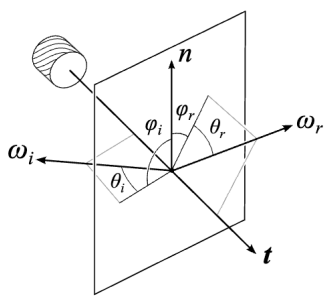
Contributor: Jesus Velarde (heyyy_sus)

Introduction

For this project, I implement the cloth/fabric rendering method referenced in the paper “A Practical Microcylinder Appearance Model for Cloth Rendering” by researchers IMAN SADEGHI, OLEG BISKER, JOACHIM DE DEKEN and HENRIK WANN JENSEN from UC San Diego. The implementation was done using light sampling ray tracing techniques and the final renders are outputted at 720p, 16spp.

Model Overview

The researchers chose to model a cloth fabric as a mesh of interwoven threads abstracted as 1-dimensional microcyliders. These microcyliders are treated as tangent vectors which represent the direction of the threads. By sampling what the tangent vector for each thread in the fabric’s “smallest patch” (that is the smallest part of the fabric that when repeated can recreate the entire fabric) and weighing the individual light scattering functions by area of the thread in the smallest patch, we can get the BRDF of the material. Below is the model and the light scattering model used:



$$L_r = \int f_s(t, \omega_i, \omega_r) L_i(\omega_i) \cos\theta_i \, d\omega_i$$

$$f_{r,s}(t, \omega_i, \omega_r) = F_r(\eta, \vec{w_i}) \cos(\phi_d/2) g(\gamma_s, \theta_h)$$

$$f_{r,v}(t, \omega_i, \omega_r) = F \, \frac{(1 - k_d) \, g(\gamma_v, \theta_h) + k_d}{\cos\theta_i + \cos\theta_r} \, A$$

$$f_s(t, \omega_i, \omega_r) = \big(f_{r,s}(t, \omega_i, \omega_r) + f_{r,v}(t, \omega_i, \omega_r)\big) / \cos^2\theta_d$$

An additional masking term is added to correctly account for grazing angles:

$$\begin{aligned} M(t, \omega_i, \omega_r) \; = \; & (1 - u(\phi_d)) \, M(t, \omega_i) \; \times \; M(t, \omega_r) + \\ & u(\phi_d) \; \min(M(t, \omega_i), M(t, \omega_r)) \end{aligned}$$

Where:

$$M(t, \omega_r) = \max(\cos\phi_r, 0)$$

The final shading equation:

$$L_{r,j}(\omega_r) = \frac{1}{N_j} \sum_t \int L_i(\omega_i) f_s(t, \omega_i, \omega_r) M(t) \cos\theta_i \, d\omega_i$$

$$L_r(\omega_r) \; = a_1 \times L_{r,1}(\omega_r) + a_2 \times L_{r,2}(\omega_r)$$

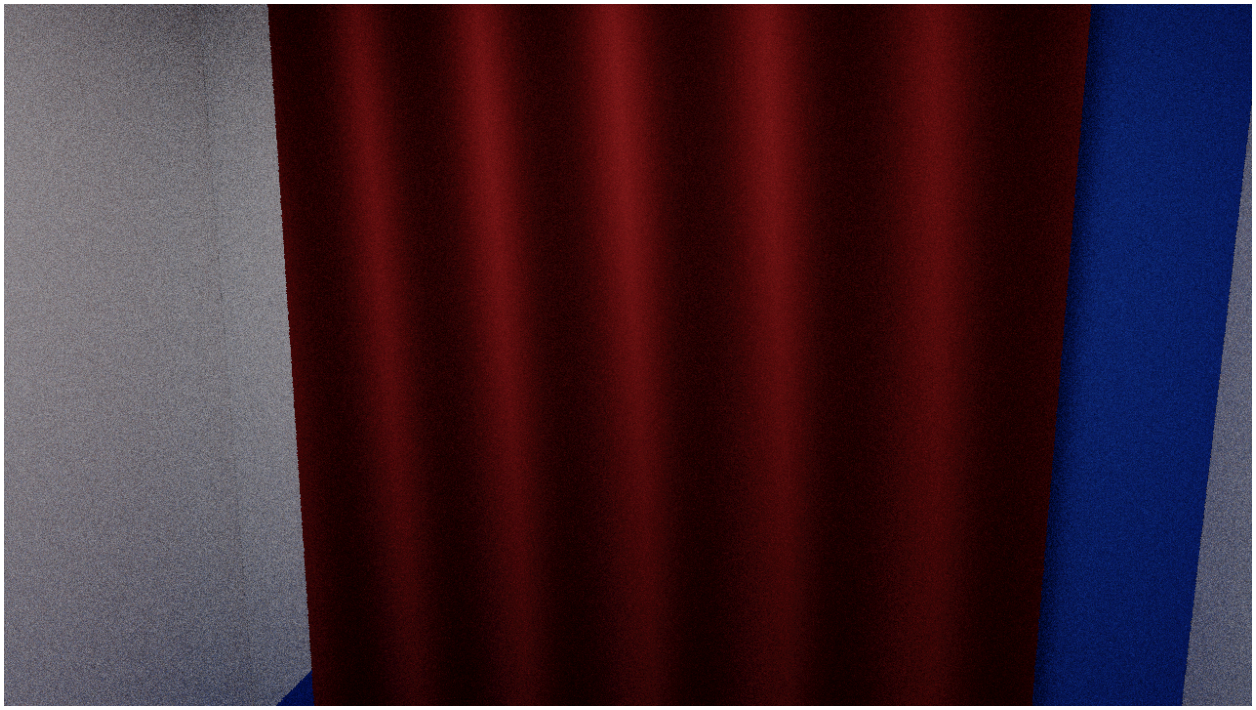
Where a_1 and a_2 are the thread area weights (does not necessarily have to sum to 1 if fabric is not water tight). The paper gives example parameters for the gamma values, k_d value, and tangent curves.

Implementation

I implemented the ray tracing in the scene.cpp:shade_pixel() function. When a cloth triangle is hit, a tangent vector t gets sampled from materials/cloth.h:sample_tangent(). Then the eval function gets called. The eval does the actual shading of the material. The individual light scattering functions are weighed and called for the tangent vector, and an additional tangent curve that is the cross of the original tangent and the surface normal. Each tangent has a different color (blue and red). This represents a fabric with red horizontal threads and blue vertical threads. The light scattering function is f_s and it implements the light scattering function said before. The primary issues I had with this function was creating an appropriate Fresnel term, correctly computing the azimuthal angles (phi), and having the reflection have appropriate brightness.

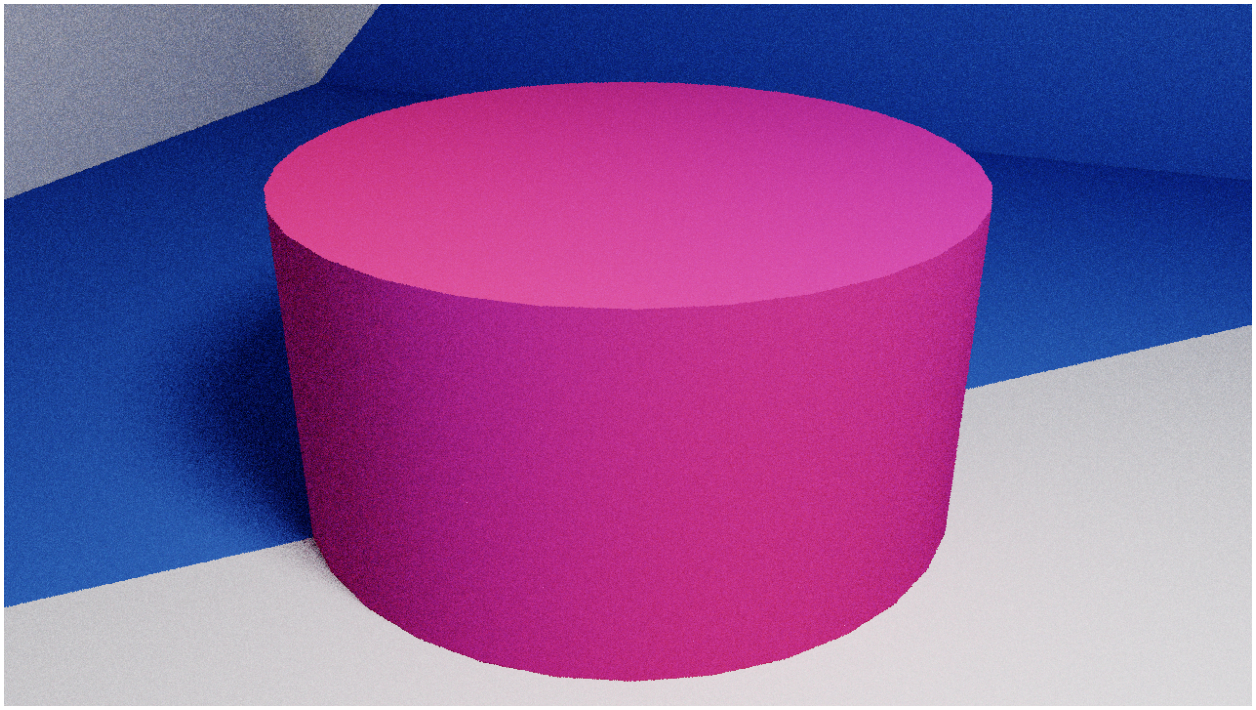
Results

I used two different scenes for the final renders: a red curtain and a cylinder with a specular fabric over it. The scenes were created from scratch and the tangent vectors were computed with a mapping function for the red curtain object along its width. The tangent vectors on the top of the cylinder had some issues.

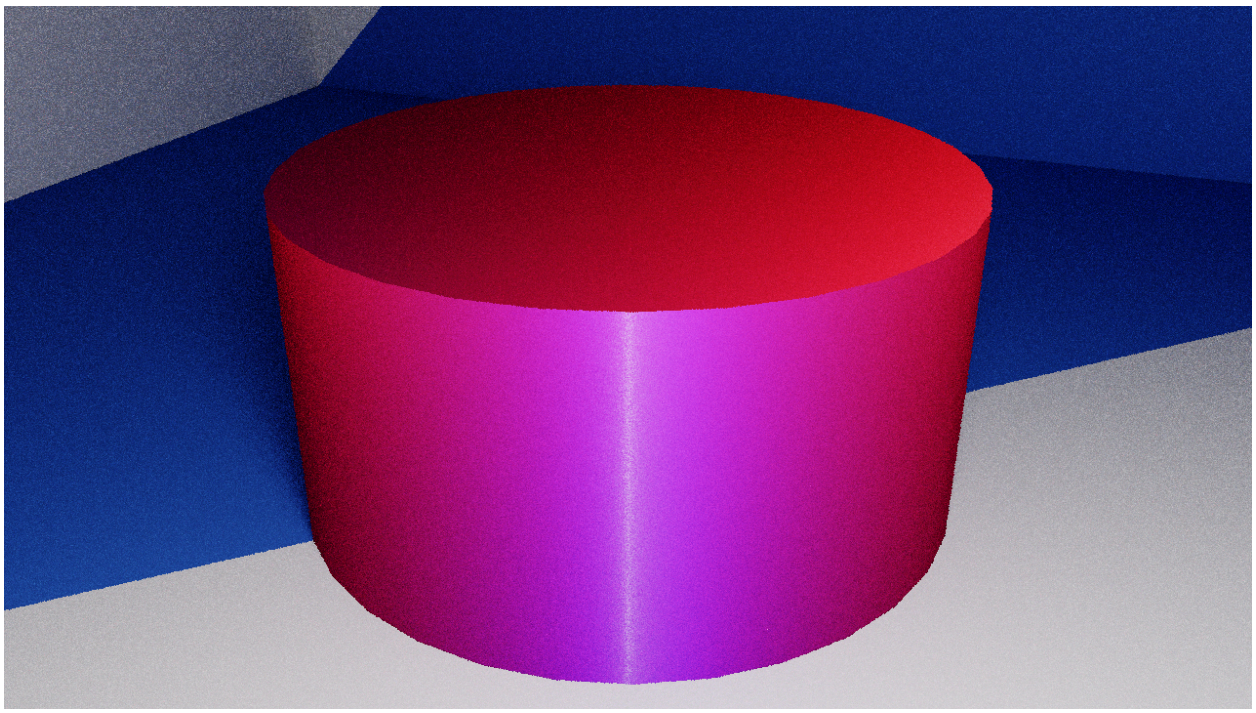


Curtain - 720p, 16spp

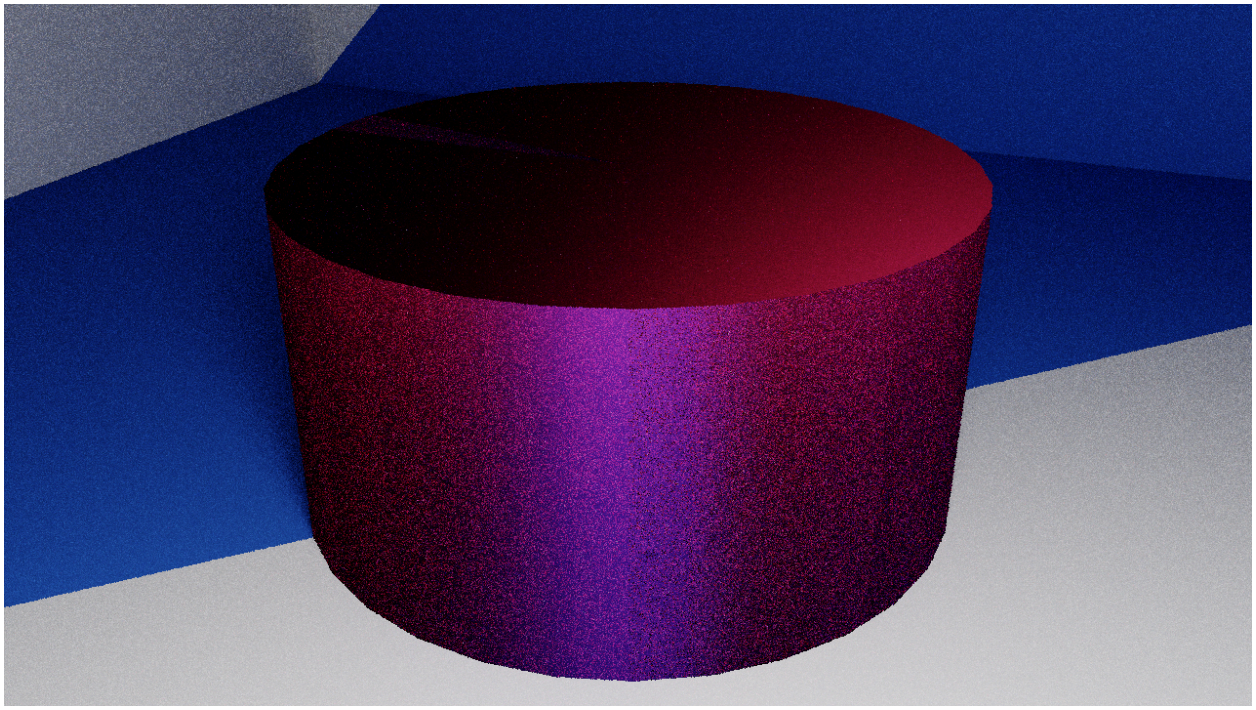
As you can see, the changing tangent vector mimics how a curtain’s changing slope reflects incoming light rays.



Cylinder without masking term - 720p, 16spp



Cylinder with masking term - 720p, 16spp



Cylinder with masking term and adjustments to albedos and fresnel term - 720p, 16spp